AD-785 410

GRAPHIC DISPLAY SYSTEM MONITOR MANUAL

Donn Bihary

Carnegie-Mellon University

Prepared for:

Air Force Office of Scientific Research
Advanced Research Projects Agency

7 July 1974

DISTRIBUTED BY:

NTIS

National Technical Information Service
U. S. DEPARTMENT OF COMMERCE
5285 Port Royal Road, Springfield Va. 22151

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AD 785 410

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFOSR - TR-74-1427

2. GOVT ACCESSION NO.:

3. RECIPIENT'S CATALOG NUMBER:

4. TITLE (and Subtitle): GRAPHIC DISPLAY SYSTEM MONITOR MANUAL

5. TYPE OF REPORT & PERIOD COVERED: Interim

6. PERFORMING ORG. REPORT NUMBER:

7. AUTHOR(s): Donn Bihary

8. CONTRACT OR GRANT NUMBER(s): F44620-73-C-0074

9. PERFORMING ORGANIZATION NAME AND ADDRESS:
Carnegie-Mellon University
Department of Computer Science
Pittsburgh, PA 15217

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:
61101D
A02466

11. CONTROLLING OFFICE NAME AND ADDRESS:
Defense Advanced Research Projects Agency
1400 Wilson Blvd
Arlington, VA 22209

12. REPORT DATE: July, 1974

13. NUMBER OF PAGES: ~~48~~ 50

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office):
Air Force Office of Scientific Research /NM
1400 Wilson Blvd
Arlington, VA 22209

15. SECURITY CLASS. (of this report): UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report):

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report):

18. SUPPLEMENTARY NOTES:

19. KEY WORDS (Continue on reverse side if necessary and identify by block number):

20. ABSTRACT (Continue on reverse side if necessary and identify by block number):

GRAPH.SYS is a PDP-11 program for use with GDP2 super-display. CDP2 consists of a PDP-11/15 computer, a STANFORD keyboard, a Hewlett Packard 1310A display scope, a start-up ROM, a line clock, a communications link to the PDP-10, 8D of 16 bit memory, and the GRAPHIC WONDER processor. GRAPH.SYS makes GDP2 into an intelligent PDP-10 graphics terminal which can simultaneously load, link and run PDP-11 programs passed from the PDP-10. This document has two parts; the first part is a manual for using the display as a terminal; part 2 is a REFERENCE MANUAL for those who wish to run programs on the PDP-11. 50

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

i.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

GRAPHIC DISPLAY SYSTEM
MONITOR MANUAL
DONN BIHARY
JULY 7, 1974

Computer Science Department
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

This work was supported by the Advanced Research Projects Agency of the Office of the Secretary of Defense under contract number F44620-73-C-0074 and is monitored by the Air Force Office of Scientific Research.

ia

GRAPH.SYS is a PDP-11 program for use with the GDP2 super-display. GDP2 consists of a PDP-11/15 computer, a STANFORD keyboard, a Hewlett Packard 1310A display scope, a start-up ROM, a line clock, a communications link to the PDP-10, 8K of 16 bit memory, and the GRAPHIC WONDER processor. GRAPH.SYS makes GDP2 into an intelligent PDP-10 graphics terminal which can simultaneously load, link and run PDP-11 programs passed from the PDP-10.

Questions, suggestions, and problems should be brought to:

BRIAN ROSEN [A630GS00]/B/A (for software OR HARDWARE)
GRAPH.XGO (for this manual)
GLSTER.XGO (for other software support).
GDP2.XGO[A210BR11] (FOR HARDWARE DOCUMENTATION)

7/7/74

This document has two parts: the first part is a manual for using the display as a terminal; part 2 is a REFERENCE MANUAL for those who wish to run programs on the PDP-11.

CONTENTS

SYSTEM START-UP

To bring up the system on the PDP-11 start the ROM by starting the PDP-11 at address 173000. This will normally load SYS:GRAPH.SYS from the PDP-10. If the user is already logged-on from the PDP-11 and has the file GRAPH.SYS on his directory, then this file will be loaded in place of the graphics system that resides on SYS. The present monitor, GRAPH.SYS, no longer has a character set. The system character set, SYS:GRAPH.GST, is automatically loaded after the monitor. If the user prefers his own character set he can name his character set file GRAPH.GST, and after the monitor is loaded his character set will be loaded in place of SYS:GRAPH.GST Minor system crashes can sometimes be fixed by a restart at address 1004 (soft restart). Major crashes can often be fixed with a restart at 1000. To be completely sure of the monitor, do a bootstrap at 173000.

A TERMINAL

Except for a more complex keyboard, the graphic system works much like an ordinary terminal. However, the addition of a PDP-11 computer allows additional features, such as intra-line editing in an SOS alter-mode fashion.

The STANFORD keyboard has 58 keys which produce 6-bit codes. These keys will be called ENCODED keys to differentiate them from the keyboard's SPECIAL keys. On both the right and left side of the the keyboard there are four SPECIAL keys; they are marked by "shift", "top", "control", and "meta". SPECIAL keys function much like the "shift" key on a ordinary teletype; when depressed they cause no immediate action, but if a ENCODED key is struck when a SPECIAL key is down, then the meaning, the PDP-11 monitor's interpretation, of the ENCODED key is altered. Each of the SPECIAL keys independently affects a unique bit in the keyboard hardware buffer. When a ENCODED key is struck, the PDP-11 monitor is interrupted and it reads a 10-bit character pattern off the keyboard hardware register. The low order 6 bits decide which ENCODED key caused the interrupt. Each of the remaining 4 bits is 1 or 0 depending upon whether or not a particular SPECIAL key is down. Now that the keyboard mechanics are understood, we can examine the way in which the software interprets each 10 bit pattern.

The keyboard can be used to communicate with the PDP-10, or with a program running on the PDP-11. Generally, if the "meta" SPECIAL key is used, then the character will be passed to a PDP-11 program (here, "character" refers to the entire 10 bit pattern from the keyboard). Communication with the PDP-10 will be considered first.

COMMUNICATION WITH PDP-10

All of the ENCODED keys have one or two engraved symbols representing ascii characters, or a name of an ascii character. When a ENCODED key is struck without any SPECIAL keys depressed, then the lower symbol represents the ascii interpretation; almost, that is, because if the symbol is an alphabetic character its lower case form is the ascii character interpretation. The "shift" SPECIAL key is used for one

purpose: to convert an alphabetic character to its upper case form. To obtain the ascii character represented by the symbol on the upper part of a ENCODED key, one must have the "top" SPECIAL key depressed when striking the appropriate ENCODED key. If both the "shift" and "top" SPECIAL keys are depressed when striking a ENCODED key, then the effect is the same as if just the "top" key was used (the "shift" key does not effect the upper symbol of a ENCODED key). Helpful note: the rubout key is called "BS", and is located on the left side of the keyboard adjacent to the letter "A". The ENCODED key "BS" is unaffected by the SPECIAL keys "top" and "shift".

The "control" SPECIAL key is closely related to the "control" key on an ordinary teletype, except that the situation is complicated because of the different ways in which the PDP-10 monitor receives characters. In normal PDP-10 ascii character mode, control characters are easily separated from text because ascii codes 1 to octal 37 are are all the control characters. The PDP-10 is also capable of entering "extended character set mode", which recognizes only the "control C" as a control character, and a special escape character. The other octal codes (control C is octal 3 and the escape is tentatively octal 27) are now interpreted as text characters. Now we have increased our character set, but we have sacrificed many important control functions. To allow both extended character set mode and control characters the escape character is used to signal the occurence of a control character. The software on the PDP-11 automatically inserts the escape character before all control characters in case the user is in extended character set mode.

To send a control character to the PDP-10 the SPECIAL key "control" is depressed while striking a ENCODED key. The SPCIAL keys "top" and "shift" still cause the actions described above, but the effect of the control key is to send the escape character followed by the low order 5-bits of the ascii interpreted ENCODED character. Because certain ENCODED characters almost always are used with the SPECIAL key "control", the PDP-11 now assumes that the SPECIAL key "control" is depressed whenever those few ENCODED characters are struck. Those keys always interpreted as control characters are: BREAK (stop output) gives control S, CLEAR (restart output) gives control Q, ALT, ESC give octal code 33; FORM (form feed),VT (vertical tab), RETURN (carriage return), and LINE (line feed) have their usual octal codes. The ENCODED character CALL is sent as a control C.

META CHARACTERS

As was earlier mentioned, the SPECIAL key "meta" signals that the character should be interpreted by a PDP-11 program. The implementation allows easy entry of numbers for parametric control of both PDP-11 monitor and PDP-11 user programs, and a simple mechanism for changing the interpretations of keys depending upon keyboard modes. The way in which meta-keys (ENCODED characters struck with at least the "meta" depressed), is not affected by the "top" or "shift" keys; The "control" key, however, usually affects what action is to be taken. A user program can, of course, access information about the state of the "top" and "shift" keys. Let us consider two classes of meta-characters: M-characters are meta characters without the "control" key depressed,

and MC-characters are meta-characters with the "control" key depressed when a particular ENCODED character is struck. Thus, for example, we have M(A) referring to the meta-character formed by striking the ENCODED character "A" with the "meta" key depressed; also we can have MC(A), which refers to the character formed by striking the ENCODED character "A" with both the "meta" and "control" keys depressed.

KEYBOARD NUMBERS

Associated with the keyboard are two variables which can be read directly by PDP-11 programs when they are interpreting a keyboard character. The numbers contained in these variables are determined by recent keyboard actions. After processing of a keyboard character (meta or otherwise), the keyboard variables are set to zero, unless the keyboard character is an M or MC digit (0,1,2,...9). The characters M(Z), and MC(Z) are reserved for resetting the keyboard variables to zero. Whenever M(-), or MC(-) is input from keyboard both keyboard numbers are set to zero and the following input number will always be negated. The two keyboard variables are used for inputting a string of digits that represent a number. Let us call one keyboard variable DECIMALVALUE, and the other OCTALVALUE. Let the symbol "*" represent any digit 0 to 9, then the actions of MC(*), and M(*) can be described by the following equations. Where "*" occurs in the equations it's value is the number digit it represents. i.e. character "2" has number value equal to 2:

OCTALVALUE←(OCTALVALUE*8) + *;
DECIMALVALUE←(DECIMALVALUE*10) + *.

In other words, typing M(a sequence of digits) or MC(a sequence of digits) will set the keyboard variables to the approiately represented octal or decimal number. If the receiving program wants an octal value it will look at OCTALVALUE. Similarly for a decimal value. The above actions of keyboard numbers is identical to the way in which numbers are input in SOS intra-line alter mode.

MC CHARACTERS

MC characters are used for putting the keyboard into a special mode which determines how M characters are subsequently to be interpreted. When an MC-character is struck a selected initiation routine is executed, and selected M-characters are given new meaning. At start-up time all M-characters are set to have no action, and the following system MC characters are available:

MC(K)	Keyboard control,	MC(E)	intra-line Editor,
MC(S)	Scroller actions,	MC(G)	Graphic control,
MC(I)	I/O - Input and output.		

i.e. typing MC(S) puts the keyboard into a mode where M characters will be interpreted by the SCROLLER module. Now if the user types MC(E), the M characters will be interpreted by the intra-line editor. The above arrangement can cause a problem of

not knowing what will happen when a particular M character is struck, since the meaning of each M character is dependent upon a previous sequence of MC characters. When such confusion arises one can simply enter the keyboard mode that he wants to be in by typing the appropriate MC character. Remember that the use of MC characters is additive - when a new mode is entered the new mode's M characters overwrite only those M characters which are having their meaning changed. To reset all M characters to no-action use M(C) while in MC(K) mode (see the following description of the keyboard MC mode). See description of module SKIT if your interested in finding out how user programs can add their MC characters to the system.

Several other MC and M characters are available for special situations. Usually these are used when an auxilliary program is caught in a loop, or the PDP-11 monitor has been garbaged.

MC("break") or M("break") - call debugging aids (DDTCAL). DDT coming soon.
MC("call") or M("call") - full system re-initiation. (i.e. restart at 1000).
MC(\) or M(\) - clear clock queue and soft restart
MC(*) or M(*) - run INITIA (system bootstrap).

KEYBOARD M-CHARACTERS

Entered by MC(K). See also the description of the keyboard module SKIT.

M(N), Normal action for non-meta characters. Useful when a user program has altered character interperpretation, or when exiting "local" mode. (see SKINOR)

M(L), Local mode for non-meta characters only. Non-control characters echo as ascii equivelents, and control characters as the 8 low order bits read directly off the hardware keyboard buffer. (see LOCAL).

M(C), Clear the M-characters of their actions (see MCLR).

M(W), turn on Space-War mode (see SKIWAR).

M(P), turn-off Space-War mode (Peace mode-see SKINWR).

M(M), setting Meta-lock software key. The keyboard number will be or'ed with all future keyboard input. (see METALK).

M(O), if zero keyboard number then turn off escape character. if non-zero then turn on escape character for output to PDP-10. (see EFCS)

THE INTRA-LINE EDITOR

The intra-line-editor facilitates editing of text through use of commands modeled after the SOS intra-line edit mode. The editor is entered by typing meta-control "E", that is, MC(E). The contents of the edit line are displayed immediately after the display of the scroller. (Actually, the first part of the edit-line may not be visible; this occurs when this part of the edit line has already been shipped to the PDP-10). The left side of the edit line is denoted by the scroller's triangular cursor. The right side of the edit line is the last visible character. Somewhere between the left and right sides of the editor there is another triangular cursor, the position-cursor. This cursor is the position where edit commands take place, and where new characters are entered. When the editor is first entered the cursors overlap, but when characters are typed the position-cursor is pushed after the last typed character. When a control character is typed, a check is made to see if it is a break character. If the control character is a break character, then the buffered line is shipped to the PDP-10 followed by the control character. If the control character is not a break character, then only the control character is sent; the edit line is not affected.

Generally, the editor is either on or off. When the editor is off, then the only editor action possible is the turning on of the editor. This is accomplished by a MC(E). The keyboard number at the time of the MC(E) is used as the size of the editor buffer; that is, is for example the keyboard number is 100, then a maximum of 100 characters can be placed in the edit buffer line. If the keyboard number is zero, the the default value is used (initially 140 decimal). If the keyboard number is non-zero then the default value is permanently changed to the value of the keyboard number.(see also the Module EDITOR for changing the default value by program control). To exit from the editor the M(E) command is used. This command de-allocates the edit buffer. If the editor is in use, then subsequent MC(E) commands will only cause the editor M-characters to be refreshed. This allows the user to type MC(S), which enters the scroller mode, make changes to scroller parameters, and then type MC(E) which re-enters the editor; this series of actions will leave the edit line un-affected allowing the user to continue his editor actions. The edit line can extend across several lines; automatic carriage returns and line feeds are inserted into the edit line (these carriage returns and line feeds are never sent to the PDP-10). The parameter UNITS controls the number of characters per line. This parameter's value can be changed when the keyboard is in SCROLLER mode, which is entered with MC(S)-see next section.

The editor meta-actions make use of the decimal keyboard number to determine the number of times to execute a particular command (where this is meaningful). In the commands below, "N" refers to the keyboard number, as described earlier, and must be entered immediately before the command. The commands SKIP, KILL, and INSERT are special because these require additional characters to be entered without the meta-key. Examples of edit commands are given after the following table of commands (all commands are M-characters except the first, which is an ordinary backspace-BS):

M CHARACTERS OF EDITOR (entered by MC(E))

BS, DELETE, Delete N characters left of cursor. Don't confuse this with the command M(BS).

M(S), SKIP, Move position cursor to N'th occurrence of the next input character. Do nothing if N occurences do not exist to right of cursor.

M(K), KILL, Like SKIP, except that all passed over characters will be deleted.

M(Line), LINE, Retrieve entire buffered line including characters that have been "burst" (see below). The edit line is emptied whenever the editor is entered, and also after typing a non-meta character after any string of control characters. This allows a line to be retrieved after a return, line, altmode, etc.

M(SPACE), RIGHT, Move position cursor N positions right.

M(TAB), FAR RIGHT, Move position cursor to end of edit line

M(BS), LEFT, Move positon cursor N positions left

M(RETURN), FAR LEFT, Move position cursor to start of buffer

M(D), DELETE, Delete N characters to right of cursor

M(B), BURST, Send all characters up to the position cursor to the PDP-10. The left cursor is moved to the current position.

M(")"), RUB RIGHT, Delete all characters right of cursor

M("("), RUB LEFT, Delete all characters left of cursor.

M(U), RUB ALL, Delete all characters.

M(Q), QUIT, Delete all, and send line-feed.

M(E), EXIT, Exit intra-line editor.

M(I), INSERT, Insert the following ENCODED characters at cursor position.

M CHARACTERS FOR SCROLLER (entered by MC(S))

The scroller displays characters coming back from the PDP-10 including echoed input characters. Commands that use a parameter use the decimal keyboard number unless otherwise indicated. The scroller now accepts variable width character sets. Two

parameters control the number of characters per scroller line. CHRS specifies the number of characters allocted for each scrolled line; CHRS is the upper limit to the number of character bytes to be entered into a scrolled line. The parameter UNITS specifies the number of "tab units" per scrolled line. For fixed width characters each "tab units" is equivalent to the number of characters. The format for variable width characters, and the meaning of "tab units" is described elsewhere (see MODULE CHR).

M(S), SCALE, Set scale to octal keyboard number (see SCLCHN).

M(I), INTENSITY, Set intensity to octal keyboard number. (see INTCHN).

M(X), X-POS, Set X-position of left margin of scrolled characters (see XCHN).

M(Y), Y-POS, Set Y-position of top of scrolled lines (see YCHN).

M(L), LINES, Set number of lines to be displayed (see LINCHN).

M(C), CHRS, Set number of characters per line. Applies to editor line too. (See CHRCHN).

M(U), UNITS, Set number of tab units per line(see UNICHN).

M(N), NEW, Reset all of the above parameters to their default values. (See SCRNEW). SCALE set to 0. INTENSITY set to 17 (full on). X-POS set to -475. Y-POS set to 475. LINES set to 56. CHRS set to 98. UNITS set to 98.

M(FORM), Clear all scrolled lines less current line. (see FORM). The current line consists of of all characters up to the last line sent to the scroller.

M(CLEAR), Like FORM, except that current line is also cleared (see CLEAR).

M CHARACTERS FOR GRAPHIC ACTION
(entered by MC(G))

These commands are for affecting the graphic processor display list. You decide what is displayed, and what is not.

M(B), BLANK, Remove scroller from display list. (See SCRBNK).

M(U), UNBLANK, Put scroller back into display.

M(R), REMOVE, Remove user graphics from display list (see USECLR).

M(P), PUT, Put the user graphics back into display list (see USERET).

M(H), HALT, Halt the entire display process. (See HALDIS).

M(G), GODISP, Start the display running again. (See GODISP).

M(D), DISPLAY, Used when the display is halted. The display will be drawn once, then repeatedly drawn "keyboard number" of times (See DISPIT).

M(C), SETCSR, The graphics will have a new CSR determined by the keyboard number. Bit 2, if set, doubles clock speed. Bit 3 will turn off wrap.(see LODCSR). For example: the sequence MC(G),M(4),M(C),M(8),M(C) doubles clock speed and causes wrap around.

M CHARACTERS FOR I/O (entered by MC(I))

The following commands affect the state the input (LAD) and output (OUT) modules.

M(O), OUTNEW, Reset all output conditions to the PDP-10. (See OUTSET).

M(T), TENLNK, Reset all input conditions from PDP-10.

M(P), PARITY, Uses keyboard number for parity control (see PARITY). (See TENSET).

SYSTEM DESIGN

SYSTEM COMMUNICATION

The system is composed of about 15 modules. Communication between modules is mainly by sub-routine calls, except for several cases of global parameter accesses. All routines are called by the standard JSR PC,XXXXXX, where XXXXXX is the routines address. In all cases the routine parameters are passed through the registers beginning with R0, and values are returned in the the registers beginning with R0. Except for R0, which is always available (a salute to BLIS!!), the called routine must protect all registers not used for passing or returning parameters.

INITIATION - MODULE INIT

Each module that requires initiation has a global initiation procedure. When the system is loaded (or restarted at address 1000) the module INIT re-initializes the system by executing the critically ordered list of initiation routines. The last routine it executes is ACTRTN, which starts up the program loop.

PROGRAM LOOP - MODULE ACTIV

The routine ACTRTN clears an "activity list" and waits for some action such as an interrupt, to place an address on the active list. When a routine is placed in the active list, it will be executed at priority level zero when the monitor has time for it. While running at priority zero, interrupts may occur. Interrupt routines perform the minimal amount of programming to ensure that no data, or vital system functions are lost (characters are buffered, and the graphic display will be re-started if necessary). Interrupts set a flag so that ACTRTN will notice that it must execute the routine. The user can place up to 3 routines in the activity list. When the monitor has time a user routine will be executed. When a routine is executed it is at the same time removed from the activity list. to enter a routine in the activity list, place its address in R0 and call the following routine (the meaning of TRAP is described in next section):

ACTVAT - TRAP 065

USE

Place a routine in the activity list. Up to 3 different routines allowed.
The routine detects attempts to put the same routine address in twice and ignores request.

PARAMETERS: R0 holds address of routine.

RETURN

R0 remains the same if successful, otherwise R0 set to zero.

ACTRTN TRAP 165 (no parameters- soft restart). Clear activities.

THE UUO HANDLER - MODULE U

The TRAP allows the user to access system routines without knowing their actual core addresses. EMT is like the TRAP except that all locations in the EMT table of addresses is reserved for the user to define. The EMT table size is set at initiation to EMTSZ. (default 20). The assembler instruction EMT is octal (104000+xxx), where xxx is the EMT number. The assembler instruction for TRAP is (104400+xxx). The trapping sequence looks just like a JSR PC,XXXXXX for all aspects of programming, where XXXXXX is the address in the TRAP table (or EMT TABLE).

TRPADD - TRAP 000, Returns address of TRAP table in R0.

EMTADD - EMT 000, Returns address of EMT table.

EMTSET - TRAP 34
USE Places user routine address in EMT table
PARAMETERS
R0 holds address of routine, R1 holds desired EMT calling number.
NORMAL RETURN
If R1 is too large for EMT table size then z-bit (in the processor) is set and no action is taken.
R0 is cleared on failure, set to 1 if successful.

EMTSZ - TRAP 141
USE Set size of EMT table (default 20). Don't use size zero.
PARAMETERS R0 holds new size (non-zero).
ACTION Old table destroyed, new allocated.

THE SPACE ALLOCATOR - MODULE SPCA

The space allocator handles all core allocations after the system is loaded. To increase program efficiency, and more importantly, to reduce program size, there is no error checking on parameters.

FREE SPACE begins immediately after the SPCA module. The FREE SPACE is divided into buffers "in use" and buffers that contain free space. Each free buffer begins with a two word header: first word is a pointer, and second word is the size of the buffer in bytes (including header words). In a "in use" buffer, there is one header word which is size of buffer in bytes (including header). Note that the buffer address that is returned from or given to the allocator routines is the address of the word immediately after the header words, and is thus the first data word.

At initiation, the space allocator assigns core for both graphics, and program data. The allocator presently assigns the same area of core for both (for 8K memory only), and

remains unconfused. The very top 3 words of cor are for a dummy buffer of zero size (at top) , a pointer to zero (1 word less), and a word to avoid merges of de-allocated core directly beneath top dummy buffer. At a low location in core there are two other headers of zero size: GRABOT, and PRGBOT. Each of these bottom buffers initially point to a single free buffer (both may be the same in 8K systems). These free space buffers together contain all the free space before the system starts creating its runtime tables. In the space allocator routine names, the middle two letters denote the type of allocator used: GR for graphics, and PG for program data.

GETSPC - TRAP 001
GTGRSP - TRAP 036

USE Retrieves free space buffer

PARAMETERS

R0 Number of words of consecutive freespace

NORMAL RETURN

Address of free space buffer stored into R0
R0 will be zero if no space available. The z-bit will be set by a test of R0 before the return.
NOTE: the allocator may occasionally give a few extra words.

BIGSPC - TRAP 002
BGGRSP - TRAP 037

USE

Allocates the largest buffer currently available

PARAMETERS NONE.

NORMAL RETURN

R0 holds address of allocated largest buffer
R1 holds size of buffer. If no space is available, then R0 will be set to zero.
A test of R0 is made to set z-bit on return.

GIVSPC - TRAP 003

USE

Returns previously allocated free space buffer.

PARAMETERS

R0 Address of buffer to be returned to free space list.

SOMSPC - TRAP 004

USE

This routine allows the user to return the top part of a previously allocated buffer.

PARAMETERS

R0 Holds address after which space no longer needed

R1 Holds address of start of buffer

NORMAL RETURN

Nothing, except that the core will not be returned, if amount was miniscule.

In order to allow de-allocation of buffers that are in the display list, the monitor has a "dead" space list. This is a linked list of those buffers that were deallocated while in the display. These buffers are returned to free space at the termination of each graphic display cycle. Two routines correspond GIVSPC, and SOMSPC.

DEDADD - TRAP 013.

USE like GIVSPC. DEDADD immediately destroys the word pointed to by R0 (the first word of the buffer). This word is used by the allocator when forming the linked list of returned space. The destruction of this value could be disastrous if it is the return word pointer for a JMS instruction. Users should not use the first word of a buffer for the return word of a JMS instruction.

SOMDED - TRAP 021. USE like SOMSPC.

Two other routines are available for the careful user. APPCLR is used to transfer the current "dead" list to another list. This new list can be deallocated any time (assuming APPCLR is called only when graphics is stopped). CLRCLR does the actual deallocation by continual calls on the routine GIVSPC.

APPCLR - TRAP 017

CLRCLR - TRAP 020

THE UTILITY ROUTINES - MODULE UTI

Only two routines here. They are used for saving an restoring registers. Register saving and restoring can be done by subroutine calls. Register values are placed on the stack during saves, and removed from the stack during restoration. The programmer must be careful to insure that the stack is at the same "depth" when restoring as when it was at save time. All registers are saved except R0.

SAVE1 - TRAP 006, and RETUR1 - TRAP 014.

LINE CLOCK SERVICE - MODULE LI

The system is interrupted every 1/60 of a second. This interrupt allows synchronization of various system actions. The routine GRAWAT is executed at interrupt time to restart the graphics, this routine is not available to the user (see GRAWAT in module GRA). When time becomes available a list of routines is executed. This list is presently fixed to a maximum of 8 routines. The system and users can put into and clear specific addresses in this list of routine addresses. The system never is using more than two of the eight positions of the queue. When the user routine is called via JSR PC,XXXXXX, register R0 holds the current system 16-bit time. This register may be destroyed during routine execution.

LINTIM - TRAP 046
- NORMAL RETURN
  - 16 - bit time returned in R0.

LINPUT - TRAP 047
- USE
  - Putting routine into clock service
- PARAMETERS
  - R0 - address of routine to be entered
- NORMAL RETURN
  - R0 set to one if successful, set to zero if no space available. If a routine is already in queue then LINPUT returns sucessful but dosn't enter the routine a second time.

LINCLR - TRAP 050
- USE
  - Removing previously put routine
- PARAMETERS
  - R0 Address of routine
- NORMAL RETURN
  - R0 set to zero if address not found within the list of routines.

ERROR TRAPPING - MODULE TRP

The module TRP's only function is to set the processor to halt on any error conditons. The user is responsible for setting up the traps for special debugging techniques.

OUTPUT TO PDP-10 - MODULE OUT

All output to the PDP-10 passes through a 300 baud (30characters/second) communications line. The PDP-11 monitor has a built-in multiplexing system for separating 7-bit character information and 8-bit image data. Eventually the PDP-10 will also have a corresponding multiplex system. The PDP-10 monitor will recognize a special escape character. The character following the escape character will be used to affect changing mode, and forcing the monitor to interpret a control character even when in extended character set mode. Extended character set mode on the PDP-10 causes all 7 bit codes to pass as text characters except for control C and the escape character. Of course, escape followed by escape will always be interpreted as escape (that is the escape octal code will be passed to program). The multiplexing on the PDP-11 is transparent to the user; the user simply makes calls on the character and image output routines and the PDP-11 monitor performs all the multiplexing. Data to be sent to the PDP-10 is processed on a first come first served basis using one large buffer to hold data until it can be shipped.

The varable CBFSZ. is used to determine the size of the buffer. Initially CBFSZ. is set to 400 octal, which means that a buffer of size 400 bytes will be allocated at start-up. The size of the buffer can be dynamically changed.

OUTSET - TRAP 161

USE

For re-setting output state. All output is stopped.
Output state is set to that at start-up time.

PARAMETERS

R0 -1 to keep current CBFSZ., otherwise CBFSZ. is set to R0.

CHARACTER MODE OUTPUT

Three routines are available for placing characters into the character output buffer. Once the character is placed in the buffer, the system will know to get around to out-putting it.

SLPONE - TRAP 31

USE

For placing an 8-bit character into the output buffer. The full eight bits will be sent, but the high order bit will be stripped off by the PDP-10 monitor.

PARAMETERS

R0 holds character in low byte.

CONOUT - TRAP 35

USE

For out-putting an escape control character. That is, the escape character is sent, followed by the character in R0.

PARAMETERS

R0 holds character in low byte.

SLPRTN - TRAP 032

USE

For outputting a 7 bit character. This routine checks to see if the eigth bit is set. If not then routine SLPONE is executed; otherwise, the eigth bit is cleared and the routine CONOUT is called. This routine is useful for out-putting characters from the keyboard since the keyboard uses this format after tranforming the keyboard character to ascii.

EXEPTION: character 221 is sent as 21 and 21 as 221. This is necessary for interfacing the escape character.

IMAGE MODE OUTPUT

Two words of data are necessary to specify an image message. The first word is the 2-byte header that always procedes the message to be sent, and the second word specifies where in core the message is found. The low byte of the header word (all words are sent low-byte high-byte by the output routine) is simply a message ID number. The PDP-11 monitor reserves the right to freely use message ID's between 1 and 77 octal. The high byte of the header has two parts. The high byte gives the number of words in the message, less the header word. This value should never get much larger than about 20 or 30. The entire message has the following format:

HEADER WORD:

bits 0 to 7: mesage ID number.

bits 8 to 15: N, the number of words in message.

MESSAGE DATA:

N words of data (N may be zero).

CHECKSUM:

one byte of data, which is minus the sum of all bytes in the header and message data. (The user can check this to be confident that the message was received correctly).

IMGRTN - TRAP 033

USE

For sending an image message to the PDP-10. The entire message is immediately copied and placed into the output buffer with the multiplexing controls. Note that the control "N" is also appended to the end of the image message.

PARAMETERS

R0: low byte holds message ID. High byte holds the size in 8 bits. NOTE that the size is in words. IF the size is zero, then only the message header is sent (i.e. the parameter in R1 is ignored).

R1: holds address of message. Beginning at that location the number of words specified in the header will be sent. NOTE: following the header word and the data words of the message, a one byte checksum is sent(-sum of all the bytes that are sent in header and data words).

INPUT FROM PDP-10 - MODULE LAD

All input from the PDP-10 is via a 4800 baud (480 characters/sec) line. The module LAD services all characters from the PDP-10. All characters off the link are buffered (up to 100 8-bit characters) until the system has time for servicing them. The service routine can be used to: send characters to the scroller (the default state); read various message types; and, in addition, the user can directly take control of the service routine to read data coming from the PDP-10.

The default state for the servicing routine is to interpret the 8-bit bytes from the PDP-10 as 7-bit ascii characters to be sent to the scroller display. The user can at any time execute the routine TENSET, which clears the input buffer and resets the link to send characters to the scroller (default mode).

TENSET - TRAP 162

USE

Full reset of state of input from the PDP-10.

PARITY - TRAP 42

R0, if non-zero flags the input to send full 8-bits to scroller.

R0, if zero (the default state) flags input to strip parity bit before sending to scroller

MESSAGES from PDP-10

In designing the message handler, several facilities were deemed important: 1) the message interpretation should be flexible - the user supplies the routine to service the data in the message; 2) assorted methods of passing data to the users routine - such as through registers, stacking, or passing the address of a buffer of inputted data (useful for graphics data); 3) arbitrary length messages; 4) optional return messages for relaying message receipt or failure of checksum etc.; 5) optional return messages for relaying a return message by the called routine. The format of a PDP-10 to PDP-11 message is (words are sent low byte,high byte):

WORD1	1
WORD2	0 (this word no longer needed)
WORD3	INSTRUCTION WORD
WORD4	CONTROL WORD
WORD5	Message size,N
WORDS 6 to WORD5+N	N data words

LAST BYTE is checksum of entire message

The INSTRUCTION word will be executed during the message handling. Usually the instruction word will be an EMT or TRAP instruction, but other possibilities are: "JSR PC,(R0)", or "JSR PC,@(R0)+" etc. This one word instruction usually passes control to a routine which returns control back to the message handler.

The control word has the format:

bits 01	PARAMETER PASSING
	BUFFER PASSING - both bits 0
	STACK PASSING - bit 0 on, bit 1 off
	REGISTER PASSING - bit 0 off, bit 1 on
BIT 2	Use graphic space allocation for buffer.
BIT 3	Register return flag
BITS 4 to 6	Number of registers to return.

PARAMETER PASSING

In BUFFER PASSING, a buffer is allocated to hold the data words. When the INSTRUCTION is executed R0 will hold the buffer's core address. Register R1 will hold the number of data words in message. The buffer will be allocated from free space; if BIT2 of the CONTROL word is set, then the free space will be taken only from graphics space. For stack passing a buffer must also be allocated to hold the data words. Prior to

INSTRUCTION execution the data words will be pushed down on the stack, register 6. Both buffer and stack passing require at least one data word. For buffer and stack passing, if no space is available for the buffer then the message will be largely ignored. Register passing does not require a space allocated buffer, and the number of data words can be from zero to six. The data words are stored into consecutive locations in the REGISTER VECTOR. The REGISTER VECTOR has word elements MR0, MR1, MR2, MR3, MR4, and MR5. Thus, if the number of data words equals 2, then the data words will be stored into location MR0, and MR1. If there are no data words, then the REGISTER VECTOR will not be affected. For all forms of parameter passing, before the INSTRUCTION is executed the hardware registers R0 to R5 are loaded from the REGISTER VECTOR. Then if buffer passing is used, R0 and R1 will be overwritten by the buffer address; if stack passing is used then the stack will be set-up. Therefore, in both stack and buffer passing, the user has control over the state of the registers when the INSTRUCTION is executed. After the INSTRUCTION is executed, the values of the hardware registers are written into REGISTER VECTOR. This allows one message to pass information to the next message through any of six registers. Also the called routine can make any changes to the parameters on the stack, or push extra elements on the stack, and in general, leave the stack pointer in a new position. After INSTRUCTION execution the stack pointer is set back to the value before the message was interpreted.

REGISTER RETURNING

If BIT3 of the CONTROL word is set, then after the INSTRUCTION execution a message with ID=22 and size specified by bits 4,5,6 of the CONTROL word, will be sent to the PDP-10. The message is really a dump of PDP-11 core beginning with MR0 of the REGISTER VECTOR. If the user does not affect the values in the REGISTER VECTOR by sending a new message, the message#22 (register return) will indicate the state of the registers after the particular message INSTRUCTION was executed.

LOADING PROGRAMS AND DATA

Blocks of data to be loaded into PDP-11 core using the following routines. The loader has one state variable, OFFSET, which is initially zero.

LOADR - TRAP 007

USE

For loading relocateable data or programs, using buffered messages.

PARAMETERS

R0, set by message handler to point to start of buffer passed from the PDP-10. Note that the message handler determines whether or not to use graphic or program space for the buffer.

ACTION

OFFSET is given the value of R0.

RELOC - TRAP 025

USE

RELOC is used (usually several times) just after a call on the routine LOADR. RELOC is called (by trapping) using a buffered message from the PDP-10. The buffer is automatically removed from core when the routine finishes.

PARAMETERS

R0,R1 set by message handler.

ACTION

The first word of the buffer is an address to point to location to begin relocation. The remaining words of the buffer are a bit pattern to denote (bit=1) which successive words to relocate. Consider the following pseudo-program:

```
INTEGER ADDRESS, BITWORD, BIT;
ADDRESS←(word 1 of buffer)+OFFSET; ! from routine LOADR;
FOR (BITWORD←(each successive word of buffer)) DO
 FOR (BIT←(each bit in bitword (left to right))) DO
  BEGIN
   IF BIT = 1 then (OFFSET added to content ADDRESS);
   ADDRESS←ADDRESS+2;
 END;
NOTE: R0 is always set to value of OFFSET.
```

LOADTR - TRAP 011

USE

Serves as an "end block" for loads. Usually called using register passing message.

PARAMETERS

R0, holds transfer address. R1, holds flags: bit 15 signals relocation by value of OFFSET to transfer address. Bit 7 says that the OFFSET should be set to non-zero value- signifying a LOADR has occurred. If OFFSET is zero and Bit 7 of R1 then no transfer takes place.

ACTION

If transter address is even then a JSR PC, to location (R0+OFFSET) is done. OFFSET is cleared.

LOADA - TRAP 010

USE

For loading absolute locations of PDP-11 core using buffered message mode.

PARAMETERS

R0, R1 set by message handler. The first word of the buffer is the address to load the data. The remaining words are the successive data words to be loaded.

SPECIAL ACTION

The data words are loaded last to first. This feature is useful when loading graphic data where the display list must always have a terminator.

LNKTAK - TRAP 41

USE Used to reset the routine which processes bytes from PDP-10.

PARAMETERS If R0 = 0 then normal monitor handling is restored. Otherwise, the address in R0 is the new routine to be called for each byte to be processed.

INTRA-LINE EDITOR - MODULE EDIT

Most of the actions of the EDITOR have been previously described in "USE AS A TERMINAL". The EDITOR has two routines that can be called globally:

BRKSET - TRAP 146

USE

A single bit in a break table will cause a control character to break. This routine sets up such a break table.

PARAMETERS

R0: BITS 0 to 17 set the break characters from octal 0 to octal 17. R1: sets the break characters from octal 20 to octal 37. E.g. if bit 5 of R0 is on, then control character octal 5 will "BREAK"; and if bit 0 of R1 is on then control character octal 20 will "BREAK". Note that bits are numbered right to left for PDP-11's. (This is the reverse of numbering for PDP-10's).

ETASET - TRAP 40

USE

Modifys ETA (Editor Transfer Address). Allows user PDP11 program to retrieve contents of edit buffer.

PARAMETERS

R0: If 0 then retsore normal sending to PDP11 (routine SLPRTN). Otherwise R0 holds address of routine to service buffer.

CHARACTER SETS - MODULE CHR

When the system is reloaded using "INITIA" (load and start at 173000), the PDP-10 sends to the PDP-11 a graphic system. After a short wait the PDP-10 monitor will send a character set. The name "GRAPH.GST" is the standard character set name. If a user is logged on, and he has a file by the name "GRAPH.GST", then this user file will be the file sent to the PDP-11 as the character set. If the user does not have a "GRAPH.GST", then the system character set "SYS:GRAPH.GST" will be loaded. The standard character set is fixed width (10 screen units) and has a line feed distance of 17 units (line feed distance includes the spacing between lines). Whenever the system is restarted, the current character set, if any, is preserved (the space allocator destroys all other buffers that were allocated from free space). Variable width character sets can also be used. Note that at the original start-up the system has a "universal null" character for each character.

CHARACTER PROCESSING

When processing a character byte, the graphics processor requires a character dispatch table-the DTBAR address. If a character byte has value "N", then the value contained in address "DTBAR+2N" is used to process the character. If this value is even (bit0 equals 0), then the value is the address of a vector description of the character. If the address is odd, then the bit zero is removed and the resulting even address must contain an interrupt routine. The only distinction that the PDP-11 monitor makes beteen "vector-described" characters and "interrupt" characters is that interrupt characters always have zero width - both for fixed and variable width character sets. The graphic monitor sets aside the top 256 words of graphic memory for the character dispatch table. The user need not know this address, however, because routines are available to affect values in this table.

TABS

Variable width characters must have the character's width, UNIT length, specified for each character. This number is usually a small integer, and must be stored in the memory word immediately preceeding the characters "vector description" (tabs need not have this value, allowing efficient coding of tab characters- see below). Besides supplying character descriptions and a character dispatch table (the table is copied by the monitor), a character set must also supply two data words: CHRUNI and CHRSTB. These two data words are located immediately before the dispatch table (thus when the monitor obtains the character sets dispatch table it first reads the data words and then copies the character sets dispatch table into the systems dispatch table). The first data word, CHRUNI, if zero,denotes that the character set is variable width. If non-zero, then the character set is fixed width with each character "CHRUNI" UNITS wide (if non-zero, CHRUNI probably wants to always have a value of 1). The other data word, CHRSTB, specifies the first tab character. Now, the last tab character is always character 367 (all character numbers are octal values). If the lowest tab character is 360, then we have the usual 8 tab characters. If the lowest tab

character is 350 then we have 16 tab characters (16 distances that we can tab). For fixed width character sets only 8 tabs are needed, since there are only 8 possible distances to the next tab character. Tabs are always performed by drawing invisible vectors. Let U be the number of character UNITS already in a scrolled line (or edit buffer), and let T be the number of different tab locations, then ((U mod T)+CHRSTB) is the tab character chosen for tabbing to the next character position.

Consider a fixed width character set with each character 10 screen units wide. Suppose the character is made with medium vectors (the standard character set has short vectors). The tab characters for such a character set would be:

```
CHR11:
CHR360:   .byte 10,0
CHR361:   .byte 10,0

  .    .    .

CHR367:   .byte 10,0
      100000        ; terminator
```

For a variable width character set with an average character width of 5 UNITS (each UNIT equal to 2 screen units) we might have:

```
CHR11:
CHR320:   .byte 2,0
CHR321:   .byte 2,0

  .    .    .

CHR367:   .byte 10,0
      100000        ; terminator
```

CHARACTER CONVENTIONS

Characters 0 to 177 are usually standard ascii characters. The null character, octal 0, is nothing but a full word terminator. Character 200 is illegal when occurring in the high byte of a character list because it signals a control word for the graphic processor. Character 200 is reserved for initializing graphics for the current character set. The character 200 should set format, and force invisible vectors. The monitor causes the character 200 to be processed before processing the scroller or edit buffer. Characters 375, 376, and 377 must always be defined as follows: 375 must be a carriage return (see below) combined with a line feed; 376 must be cursor (whatever cursor you like, but it should start and end at the same place); and 377 must be the combination of characters 375 and 376. Character 370 is the blink

character, and character 371 is the unblink character. Characters 372 to 374 are reserved for expansion. The characters immediately below 370 are used as tab characters. Carriage returns are performed using a SETX control word. Since the monitor must be able to dynamically change the value of the carriage return position, the SETX control word must always be at the same position in all characters with a carriage return (characters 15, 375, and 377). For now the SETX control word is the second word of the character, the first word should be a zero.

CHARACTER ROUTINES

CHRSET - TRAP 153

USE

Allows user to make single changes to the character dispatch table.

PARAMETERS

R0: holds new value for dispatch of character. If R0 is zero, then the universal null character is used for the character dispatch. If R0=-1 then do not replace the character; only return address of character r1 in R0.

R1: holds the character whose value is to be changed.

RETURN

On return, register R0 will hold the value that was previously in the dispatch table.

CDSCLR - TRAP 154

USE

Clears the dispatch table to all nulls.

PARAMETERS (none)

CDSSET - TRAP 155

USE

This routine copies the data words and character dispatch table. After copying, the user should deallocate his own copy (the system can do this for you- see below).

PARAMETERS

R0: holds address of core allocated from free space for the entire character set description. This value is saved until core is released, and is also used by the space allocator for preserving a character set during a restart.

R1:Holds dispatch table address. Note that the two words preceeding the dispatch table must be values for CHRUNI, and CHRSTB,respectively.

RETURN

On return, R0 holds the system dispatch table address and R1 holds the address of the dispatch table copied from user.

CHRDEL - TRAP 157

USE

This routine deletes the current character set by Writing the universal nulls into the character dispatch table, and then deleting the character description segment.

PARAMETERS (none)

The most likely method of passing a character set to the PDP-11 is by loading a completely new description from the PDP-10. The following special routine makes loading a new character set very easy. Simply prepare a character set using the format below, and then compile the program using MACX11, and linked using LINK. If the program is relocated to zero (using the "M" AND "G" swicthS in LINK), then the routine will correctly relocate the dispatch table.

CHARACTER SET FORMAT

1) Character set vector descriptions. Also any "interrupt" characters.
2) CHRUNI value.
3) CHRSTB value.
4) The dispatch table. Label the first location DTTAB. Remember that all 256 characters must be given a reasonable value. (repeat your null character's address if nothing better).
5) Now have another label CHRSTR. Following the label CHRSTR we have the following short program:

```
CHRSTR:
      MOV #DTTAB,R1 ; R0 already holds relocation value
      104556             ; this is trap 156- CHRLOD
      RTS PC
.end CHRSTR         ; end of program- transfer to CHRSTR
```

CHRLOD - TRAP 156

USE

This routine facilitates loading of a new character set. Use this routine with a just loaded module containing a character set description with the above format.. Any previous character is deleted. The new dispatch table is copied. After that, all locations beginning with the dispatch are returned to free space.

PARAMETERS

R0: This must hold the address that the space allocated for the character set descriptions. Note that the loader automatically supply this value.

R1: Must hold the character dispatch table address (before relocation, if PDP-11 supplied relocation).

NOTE: Before sending a character set, a message should be sent to delete the old character set. Also the scroller should be cleared. These two operations will insure that a

large free segment will be available for the space allocator to pass to the newly arriving character set. Also, for convenience, several messages should be tacked onto the end of the character set. These messages should contain information about the number of lines in scroller, UNITS per line, and other related information peculiar to the new character set.

GRAPHIC DISPLAY - MODULE GRA

The module GRA controls all graphic displays. The GRA module sets up a sequence of graphic instructions to be executed. Three instruction locations are reserved: one for the scroller, one for the editor, and one instruction for the user to play with. The only reasonable instructions that can be put into these slots are XQT's and JMS's. With the JMS instruction, the user can effectively set up an arbitrarily complex display list. The following routines are used to affect what is placed into the above three locations.

SCRCLR - TRAP 071

USE

Removes instruction in scroller position.
(I. E. a jump to the next location is put in).

SCRDIS - TRAP 073

USE: Insert an instruction into scroller position.
PARAMETERS: R0, holds value to put in.

USECLR - TRAP 072

USE

Remove instruction from user position, and save the value.

USERET - TRAP 144

USE

Return the value previously in the user position. This value is the value that was removed by the routine USECLR.

USEDIS - TRAP 074

USE

For putting an instruction into the user position.

PARAMETERS

R0: holds value to be put into the display. Thus, R0 should be an address with the opcode bits for a JMS or XQT (see hardware doc.).

EDCLR - TRAP 075, Removing instruction from editor position.

EDDIS - TRAP 076

USE

For putting a value into editor position.

PARAMETERS

R0: holds the value to be inserted into the instruction sequence.

GRAADD - TRAP 027

USE

R0 holds address of pointer to the location of the instruction (graphic JMS) for scroller sub-display.

(R0+2) is address of the location of pointer to the instruction (graphic XQT) of the line editor.

(R0+4) is address of pointer to location of the user sub - picture instruction (graphic XQT or JMS).

Three variables govern graphic display. The variable SIC is 1 when graphics is running. When the display list is fully processed, SIC is set to 0. TIC is a count of the number of tics of the line clock that have occured since graphics was last restarted. If this number grows to 20, then the monitor assumes that the display list is being quite unreasonable. Graphics will be restarted, unless the variable GODIS is zero. GODIS, if negative, means that the user wants the graphic processing to be starting at each appropriate time (in sync with line clock). If GODIS is positive, then each time the graphics is re-started GODIS is decremented. If GODIS is zero, then graphics will not be restarted.

GRASTP - TRAP 100, Halt graphic display instantly. Also set GODIS to -1.

DISPIT - TRAP 132

USE Set start up of graphics.

PARAMETERS

R0:The value of R0 is given to GODIS. If GODIS results in non-zero then graphics is restarted.

HALDIS - TRAP 130, Soft stop of graphics. GODIS is set to zero.

GODISP - TRAP 131, Slow start of graphics. GODIS set to -1.

LODCSR - TRAP 012

USE

Affecting the CSR register of graphics.

PARAMETERS

R0: bit 2- turn off clock divide. Bit 3, turn off wrap.

The control word interrupt traps to address 104. This interrupt is serviced by the monitor. The monitor extracts the 4 bit operand and selects a routine from a 16 word table of routine addresses. The called routine can use registers register R0. Graphic display "pauses" during an interrupt. When the routine returns (RTS), the graphics will be continued from the pause state if R0 is non-zero. The interrupt routine should be very short, and should not make calls on any monitor routines that could cause "race" conditions. Currently, interrupt 0 turns off intensity every other 1/4 second. Interrupt 1 turns the intensity on to the value obtained in the last blink interrupt. Interrupt 1is "unblink". A control word interrupt that has not been given a "meaning" will cause a trap at address zero.

INTRST - TRAP 015

USE

Put a routine into the interrupt dispatch table.

PARAMETERS

R0, holds the number of the routine. This value should be between 10 and 17 octal. (codes 0 to 7 reserved for system). R1, holds address of interrupt routine. Note: currently if the graphic processes an undefined interrupt then the PDP-11 will HALT.

BLINKING

The interrupt operand 0 causes BLINK-ON, and the operand 1 causes a BLINK-OFF. These control words must always come in matched pairs with the items to be blinked laying between the control words. Blinking is accomplished by setting intensity zero when BLINK-ON occurs. The intensity is restored to its previous value with the BLINK-OFF character. Blinking characters also exist, see BLINKING CHARACTERS.

STANFORD KEYBOARD - MODULE SKIT

The stanford keyboard has previously been discussed in terms of actions by the computer terminal user. The following discussion is concerned with programming interactions with the user's programs on the PDP-11.

When a key is struck the monitor is usually the program to notice that a character is waiting to be serviced. However, the user can obtain control of the very lowest level

of interrupt handling. The keyboard has a standard interrupt vector at location octal 200. To obtain the keyboard character directly, the user needs only to place an interrupt routine's address at this location. When the user wishes to return keyboard handling to the monitor one of the following routines may be chosen:

SKINWR - TRAP 063
SKIWAR - TRAP 023

USE

Reinit the keyboard status and interrupt vector. The only differrence between SKIWAR and SKINWR is that SKIWAR will set a bit in the status word that causes an interrupt to occur each time the bit pattern from the keyboard changes. Thus in "war" mode, depressing the character "A" will cause an interrupt, and releasing the key "A" will cause another interrupt. NOTE: the "war" bit is occasionally called the "space war" bit; do not confuse the bit with the game of space war. Also as a side affect to both of the routines: any currently pending character is thrown away.

In case you do write your own keyboard routine, beware that the characters coming off the buffer (165200 for status, and 165202 for the buffer) do not correspond to ascii characters. The bits in the hardware buffer have the following format (from left to right):

"MOOOOIACTSEEEEEE"

M bit: On, if meta key is depressed.
O bits: These bits are always zero
I bit: Service at interrupt time (use very carefully).
A bit: This bit is zero, but when the routine SKIRTN interprets a keyboard character this bit will (if set) stop keyboard translation to ascii. (processing characters form PDP-10).
C bit: On if control key is depressed.
T bit: On if top key is depressed.
S bit: On if shift key is depressed.
E bits: These bits are the or'rd value of all endoded keys that are depressed. (See table pg. 48)

The bits of the status register correspond exactly to the bits for a standard DEC teletype reader status, except that bit 0 is used to signal "war" mode (any change in the hardware buffer causes an interrupt if bit 0 is on).

In order to translate the low byte of the keyboard buffer (TSEEEEEE) into an ascii equivelent, just execute the following routine:

KEYASC - TRAP 064
USE: Transform keyboard value to ascii.
PARAMETERS
RO holds keyboard bits to be decoded. The high byte is ignored.
RETURN: RO holds the ascii equivelent.

The routines SKIWAR and SKINWR are not necessary to gain rather complete control control of keyboard input. Rather the mechanism of "meta" characters should allow an adequate PDP-11 program to keyboard interface.

The first action to occur during a character interrupt (with the monitor's routine in control) is a check for a special interrupt character. These were previously described in "USE AS A TERMINAL", but are repeated below.

MC("break") or M("break") - call debugging aids (DDTCAL)
No DDT module available at this time.
MC("call") or M("call") - full system re-initiation
(restart at 1000).
MC(\) or M(\) - clear clock queue and soft restart

MC(break) and M(break) are useful only if the following routine has been executed:
DDTCAL - TRAP 133
USE
Pass the address of a debugging routine. This routine will be JSR'ed to when the M("D") or MC("D") character is struck.
PARAMETERS: RO holds address of debug routine.

FAKING KEYBOARD INPUT

Two routines are available for sending input to the keyboard handler. The routine SKIRTN is the same routine that handles actual keyboard input.

SKISOM - TRAP 126

USE

Allows a buffer of 16 bit characters (in keyboard input format), to be interpreted by SKIRTN. In particuly, the SOS "Z" command can be implemented with this routine.

PARAMETERS

R0: contains address of buffer.

R1: contains number of words to process.

ACTION

Each word in the buffer is processed using SKIRTN.

NOTE

When inputting a buffer from the PDP-10, the loader (coincidently?) sets up R0 and R1 correctly when using buffered message handling.

KEYBOARD STATE

The actions of the routine SKIRTN must be postponed until after a discussion of the keyboard's "state". The keyboard has associated with it a state. The state includes the keyboard numbers, the keyboard transfer address(KBTA), the MC table and the M table. The keyboard numbers were previously discussed (see KEYBOARD NUMBERS). The KBTA is a variable which holds the address of the routine that is to process all non-meta characters. At system start-up the default routine is SLPRTN (this is a routine common to the output module OUT). Two other routines often are used for the KBTA: LOCIT which is the routine that outputs to the scroller keyboard characters (somewhat like local mode on a TTY); the other routine is the routine which services the non-meta characters during edit mode. Four routines are available for affecting the value of KBTA.

SKINOR - TRAP 060, Set the value of KBTA to default value SLPRTN.

LOCAL - TRAP 022

USE

Put keyboard into local mode by placing address of routine LOCIT into KBTA.

SKIGET - TRAP 056, Return value of KBTA in R0.

SKIPUT - TRAP 057

USE: Changing the value of KBTA.

PARAMETERS: R0: R0 is the new value given to KBTA.

EFCS - TRAP 163

USE

Control use of escape character. Ascii output of control characters to

PDP-10 will not be preceeded by escape (control P), if R0 is set to 0. If R0 is non-zero then a control P will be used. Note this command does not affect control characters for image mode.

AFFECTING META TABLES

Meta tables are the principal means of communication between a PDP11 program and the keyboard. Both M and MC tables contain addresses of routines that will be called when their associated meta characters are struck. At system startup the M and MC tables are cleared to zeros, then the standard system MC characters are copyed into the MC table. When an MC character is struck (before the routine associated with it is called) the keyboard handler copys a table of M characters into the M table. The table copyed in is called the MCMR table, where MCMR stands for "Meta-Control Meta Read". The format of the MCMR table must be (where all characters are ENCODED characters and not ascii):

```
TABLESTART:         ; example of MCMR table
    E1          ; first character
      E1ADDR    ; address of routine  for this M character
      E2
      E2ADDR

      EN'th
      ENADDR
      .  .  .

      0
                    ; note that the 0 signal the end of table
```

Rather than keep an address of the MC routine and the the address of its own M table, it has been decreed that the word preceeding the first word of the routine (MC routine address -2) must contain a pointer (an address) to the start of its MCMR table. The method of setting up MC characters is probably still confusing to the reader. Try reading the descriptions of the following routines. The mechanism is actually very simple....

MCSET - TRAP 061

USE: Setting up a MC character.

PARAMETERS

R0:The Keyboard encoded character that will cause the MC action. E.G. if R0 is 1, then the character "A" will be an active MC character. NOTE that all of the upper case ascii letters corresspond to the keyboard encoded letters (by coincidence).

R1: contains address of routine to be called. Note that immediately preceeding the routine must be the address of a table of M-characters .

ACTION:

Very little. The value supplied in R1 is put into the the appropriate MC table location (as determined by R0).

NOTE:

If R1 contains zero, then no action will take place when the particular MC character is struck.

MCLR - TRAP 024

USE

This routine clears the M-Table of all values. Thus, M characters will cause no monitor actions.

SKIRTN - TRAP 055

USE

SKIRTN is routine which processes all keyboard characters.

PARAMETERS

R0: holds the keyboard character to be interpreted.

ACTIONS

If the meta bit is not on (bit 15 of R0), then the low byte of R0 is usually converted to ascii using the previously described routine KEYASC.
To stop conversion to ascii simply set the "A" bit, bit 9.

PROCESSING NON-META CHARACTERS

When processing a non-meta character, the routine SKIRTN sets up register R0 to contain the ascii value of the input character in the low order 7 bits of R0. If the input character to SKIRTN is a control character, then the 7'th bit (8'th bit from left) of R0 will be turned on. Note also that bit 6 is always chopped off on control characters. After setting-up register R0, the routine whose address is contained in variable KBTA (defalt SLPRTN sends the character to the PDP10) will be JSR'ed to. This routine is free to use all the registers (R0 to R5). The user can access the original input character by examining register R1.

PROCESSING META CHARACTERS

Firstly, for all types of M and MC character processing, when a routine is called by the control of SKIRTN, all registers will be available for use by the called routine. In particular the registers will contain the following information:

R0: The original input character to SKIRTN.
R1: The decimal keyboard number.
R2: The octal keyboard number.
(in the present implementation R3, and R4 also
hold the octal and decimal keyboard numbers. This will
not be true of later versions).

DESCRIPTION OF THE CONTENTS OF R0:

THE CHARACTER CODE FOR THE KEY STRUCK IS STORED IN THE LOW ORDER BYTE OF R0. BITS 6 AND 7 REPRESENT TOP AND SHIFT. BITS 0-5 ARE THE CHARACTER CODE ACCORDING TO THE FOLLOWING TABLE:

OCTAL	KEY	OCTAL	KEY	OCTAL	KEY	OCTAL	KEY
1	A	21	Q	41	BREAK	61	1
2	B	22	R	42	ESC	62	2
3	C	23	S	43	CALL	63	3
4	D	24	T	44	CLEAR	64	4
5	E	25	U	45	TAB	65	5
6	F	26	V	46	FORM	66	6
7	G	27	W	47	VT	67	7
10	H	30	X	50	(	70	8
11	I	31	Y	51	)	71	9
12	J	32	Z	52	*	72	:
13	K	33	RETURN	53	+	73	;
14	L	34		54	,	74	BS
15	M	35	LINE	55	-	75	ALT
16	N	36		56	.	76	
17	O	37		57	/	77	
20	P	40	SPACE	60	0		

BIT 6:	1 IF SHIFT	0 IF NO SHIFT
BIT 7:	1 IF TOP	0 IF NO TOP

M character processing is simple. Check the M table location for the input character. If this value is zero then no action. Otherwise the value is taken to be an address to JSR to. MC character processing is similar, except that before the routine is JSR'ed to, the MCMR table is read into the M table. This reading of the MCMR table is the only way in which addresses can be placed into the M table. When communicating with a program on the PDP-11 it would sometimes be desirable to have a "meta-lock" key; that is, when meta lock is on, then all software processing would assume that the meta key was actually pressed when any encoded key is struck.

METALK - TRAP 325

USE

Locking and unlocking software meta-lock bit and other bits.

PARAMETERS

R0: value of R0 will be or'ed with all future input from keyboard.

EXAMPLE FOR SETTING UP A MC-CHARACTER

Suppose we wish to define the MC-character "H". Pretend that "H" is mnemonic for the routine HELP. We also must create the MCRM(H). This table describes which M-characters will be affected when MC(H) is struck. Suppose we have routines AHELP, BHELP, and CHELP that we wish to have called when M(A), M(B), and M(C) are struck,respectively. The table would be:

```
BIND MCMRTH=PLIT("A",AHELP,"B",BHELP,"C",CHELP,0);          ! in bliss

MCMRTH:                         ; now for assembler
        'A
        AHELP
        'B
        BHELP
        'C
        CHELP
        0
```

Now the routine HELP would look like:

```
INLINE("    MCMRTH    ; table address");
ROUTINE HELP=
  BEGIN blah blah blah END;
```

or in assembler:

```
      MCMRTH
HELP:
        blah
        blah
        blah
        RTS PC
```

To ready the MC(H) character we must put the address of routine HELP into the MC table. This can be done as folows:

```
R1←HELP        ; ! the address of routine
VREG←"H"               ; ! this is ascii (works for alphabetic)
TRAP('61)              ; trap to the MCSET routine
```

or in assemble..

```
MOV *HELP,R1   ; address of routine
MOV 'H,R0
TRAP 61        ; trap to routine
```

And now, each time MC(H) is struck, the following actions will take place:

1) The routine HELP will be called with the registers having the following values:

2) Th addresses of routines AHELP,BHELP, and CHELP will be copyed into the M table locations for codes 1("A"),2("B") and 3("C"), respectively.

R0:	Bit pattern for MC(H)
R1:	Decimal keyboard number.
R2:	Octal keyboard number.

THE SCROLLER

Unlike a TTY in which a continuous record of output is kept, the graphic monitor maintains a display list of characters returned from the PDP-10. This display list is composed of lines of characters begin at the top of the screen. As new lines are entered at the bottom of the screen the lines on the top are removed. The scroller is line oriented. The parameter CHRMAX holds the maximum number of characters per line. Whenever a new line is started CHRMAX is used in deciding how big the display buffer for that line must be. Characters are entered into the "current" line using the routine SCRRTN. As each character is entered into the current line, the number of units (see CHARACTER SETS) is kept. This is necessary to proper tabs. The

parameter UNIMAX is the maximum number of UNITS that are allowed in one line. A new line, that is a new buffer, is allocated for one of three reasons:

1) Line feed. Each occurence of a line feed terminates the current line and a new buffer is fetched.

2) Unit overflow. Whenever the number of UNITS in a line exceeds UNIMAX a carriage return/linefeed is inserted, and a new buffer is fetcheo. UNIMAX is used to make the line finish at the same place on each line (for variable width too).

3) Character overflow. Whenever the number of characters in a line exceeds CHRMAX a carriege return/line feed is inserted and a new buffer is fetched. Note character overflow signals that the user should increase the value of CHRMAX.

The following routine is the only routine for entering characters into the scroller.

SCRRTN - TRAP 120

USE

Entering 8-bit characters into scroller.

PARAMETERS

R0: Low byte of R0 holds byte to be entered.

SCRUNB - TRAP 107

USE

Puts the scroller back into the display list.
(see SCRCLRi, TRAP 071, for complementary action).

CLEAR - TRAP 134

USE

Removes all scrolled characters from the display. The screen is cleared.

PARAMETERS (none)

FORM - TRAP 104

USE

Works like a CLEAR except that the "current" line is left in the display.

There are seven basic parameters to the scroller. The following table shows both the default values and the usual range in parentheses. Note that some parameters are entered by octal keyboard number. Other values are decimal.

XCUR	Carriage return position. Default -475. (-511)
YCUR	Top of screen. Default 475. (to 511).
SCALE	The scale for graphics. Default 0. (0 to 17 octal).
INTENS	The intensity for graphics. Default 17. (0 to 17 octal).
LINES	Number of lines allowed in scroller.Default 56. (0 to 56).
CHRS	Number of characters per line. Default 98.
UNITS	Number of units per line. Default 98.
JUMP	Number of lines to jump when scroller reaches max. Default 0.

SCRNEW - TRAP 111

USE

Sets all of the above parameters to the default value.

Each of the parameters can be set individually. The following table describes 7 separate routines.

ROUTINE	TRAP	PARAMETER AFFECTED
XCHN	112	XCUR
YCHN	113	YCUR
SCLCHN	115	SCALE
INTCHN	114	INTENSITY
LINCHN	116	LINES
CHRCHN	117	CHRS
UNICHN	101	UNITS
JMPCHN	016	JUMP

SCRDEF - TRAP 030

USE

For setting the defaults for the above 8 parameters.

PARAMETERS

R0: holds the address of 8 word buffer of values to be given to the above 8 parameters, respectively. I.e. first word changes value XCUR, and second alters YCUR; and so on.

INDEX

INDEX